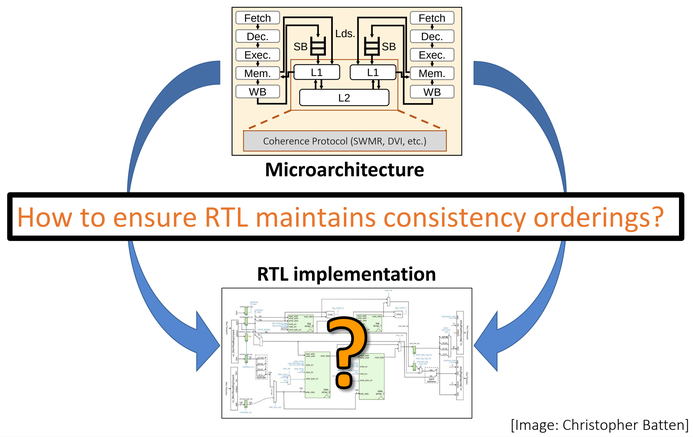
RTLCheck: Verifying the Memory Consistency of RTL Designs

Yatin A. Manerkar, Daniel Lustig*, Margaret Martonosi, Michael Pellauer*

PRINCETON UNIVERSITY

***NVIDIA**

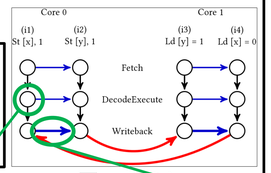
Motivation



From Microarchitecture to RTL

Axiomatic Microarchitectural Verification

```
Axiom "WB_FIFO":
forall microops "a1", "a2",
(OnCore c a1 /\ OnCore c a2 /\~SameMicroop a1 a2
  /\ ProgramOrder a1 a2) =>
    EdgeExists((a1,DX)), (a2,DX))) =>
       AddEdge((a1,WB)), (a2,WB))).
```

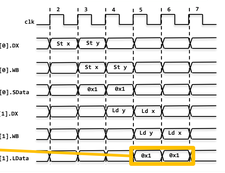


Nodes == µarch. events

Edges == local happens-before relations between nodes

Temporal RTL Verification (using SVA, etc)

```
assert property @(posedge clk) (
    if (Core[1].PC_DX == 4) then
        in next cycle (Core[1].Ldata == 1))
    end
)
```



Concrete signal values and clock cycles

Note: Property abstracted for brevity

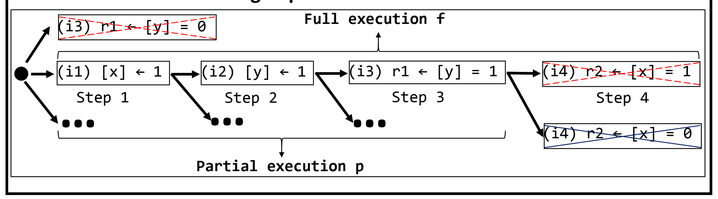
The Axiomatic-Temporal Mismatch

- Axiomatic models check executions as a unit
- Axioms applied to an entire execution
- **Easy and efficient outcome filtering**

mp (Message Passing)

Core 0	Core 1
(i1) [x] ← 1	(i3) r1 ← [y]
(i2) [y] ← 1	(i4) r2 ← [x]
Outcome: r1 = 1, r2 = 0	

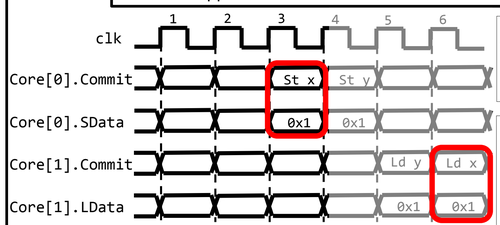
- Temporal models generate executions step-by-step as a "**tree**"
- Eliminating executions by outcome requires **expensive global analysis**
- SVA verifiers **approximate** by **only checking constraints up to current step**
- **Makes outcome filtering impossible!**



Lack of Outcome Filtering in SVA

Simplified property (using outcome filtering):

```
BeforeAllWrites:
If Load reads value 0, then
  Load happens before all stores to its address
```



After 3 cycles:
1. Load hasn't occurred yet!
2. St happens before Ld!

Counterexample!

After 6 cycles:
1. Load does not read 0

No Counterexample!

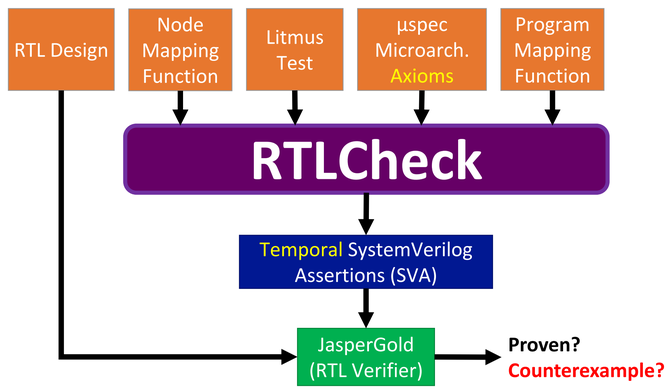
But verifiers don't check future cycles!

Solution: Translate all cases, keep track of **load value constraints** (data restrictions for a given edge)

Note: Axioms/properties abstracted for brevity

RTLCheck Block Diagram

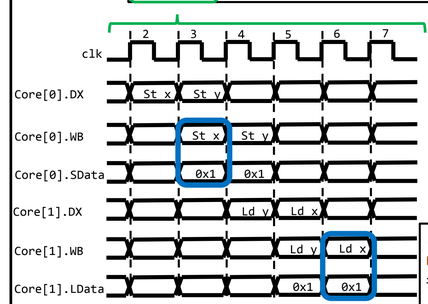
Mapping Functions translate µarch primitives to their RTL equivalents.



Mapping µhb Edges to SVA

How to map the edge `Ld x=0 @ WB` —hb→ `St x @ WB` ?

Naïve attempt: `##[0:$] <Ld x=0@WB> ##[1:$] <St x@WB>`



Unbounded ranges match too many cycles!

Solution:

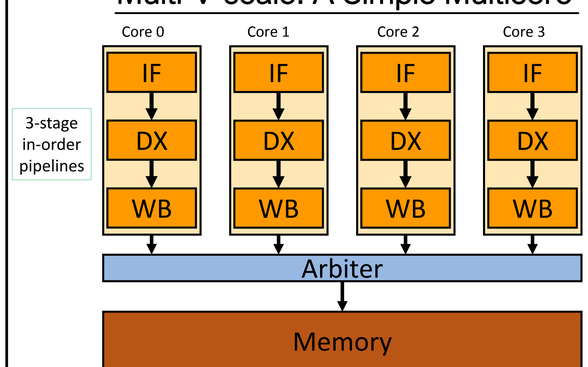
src —hb→ dest

Maps To

```
(~(mapNode(src, None) ||
mapNode(dest, None))) [*0:$]
##1 mapNode(src, lc) ##1
(~(mapNode(src, None) ||
mapNode(dest, None))) [*0:$]
##1 mapNode(dest, lc)
```

`##[<x>:$]` = x or more cycles delay
`<x>[*0:$]` = 0 or more reps of x
`##1` = Delay of 1 cycle

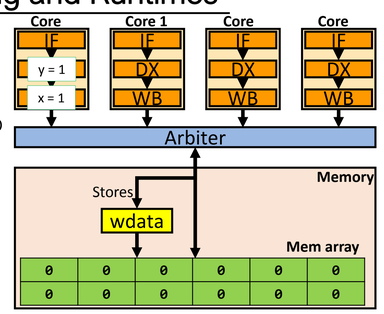
Multi-V-scale: A Simple Multicore

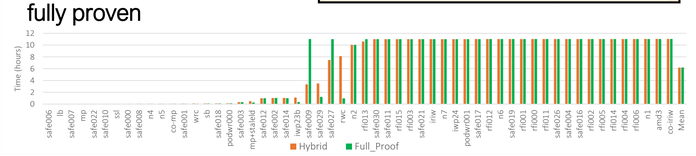


- Multicore version of RISC-V V-scale processor
- Mem ops send addr to mem in DX, and recv/send data to mem in WB
- Cores access read-only instruction memory (not shown) in parallel

V-scale Bug and Runtimes

- RTLCheck discovered bug in V-scale's memory implementation
- When two stores are sent to mem in successive cycles, first store is dropped
- Runtime to verification below (max. 11 hrs/test)
- Avg. runtime 6.2 hrs/test
- Up to 89% of properties fully proven





Conclusions

- RTL must be verified against formal specifications of MCM behaviour
- **RTLCheck:** Automated MCM verification of ***arbitrary*** RTL against ***arbitrary*** microarchitectural orderings
  - Translates microarch. axioms to equivalent temporal SVA properties
  - Can handle arbitrary ISA-level MCMs (SC, TSO, ARM, …)
  - JasperGold proves most of generated properties in minutes or hours
- RTLCheck enables full-stack HLL-to-RTL MCM verification (with rest of Check suite) across a collection of litmus tests